Raphaël DI MASCIO Hugo VERNEUIL

Lisezmoi

Ce document traite de la SAE5.3 : **Modélisation mathématique**
Reconnaissance faciale en temps réel. Y figurent les actions effectuées, les difficultés rencontrées ainsi que les pistes d'amélioration envisagées par le binôme Raphaël D. et Hugo V.

Lien du depot git : https://github.com/RaphaelDiMascio/SAE5.3
Rappel consignes: voir document "Consignes.md"

Exercice 2:

L'exercice 2 dont nous rappelons l'intitulé : "*Adaptez le 2ème notebook à d'autres algorithmes de classification de votre choix, tels que la régression logistique, les arbres de décision, etc.",* n'aura pas posé de difficulté particulière mais nous aura permis de bien comprendre le fonctionnement du programme et la place de chaque composante au sein de celui-ci, après un petit peu de débogage. Ainsi, nous avons pus implémenté le programme grâce aux algorithmes :

- de Régression logistique : "2_algo_regression_logistique.ipynb"
- d'Arbre de décisions : "2_algo_arbre_decision.ipynb"
- de Gaussian Naives Bayes : "2_algo_GNB.ipynb"
- de Perceptron: "2_algo_Perceptron.ipynb"

Exercice 3:

Il est à noter qu'à ce stade de la SAE, nous avons décidé d'augmenter le nombre d'image du dataset de 10 par visage à 30 par visage, ceci afin d'augmenter la précision des modèles, nous n'avons pas constaté d'impacte notable sur les performances en termes de ressources et de rapidité des algorithmes.
L'exercice 3, dont voici l'intitulé : "*Essayer d'adapter votre système pour le cas d'utilisation de reconnaissance binaire suivant: le système devra répondre* `admis` *ou* `non admis` *suivant que le visage détecté est le vôtre ou non.*" nous aura demandé beaucoup plus de travail comparativement à l'exercice 2.
Son fonctionnement peut être résumé ainsi : une variable stocke le label du visage désigné comme "Admis" (visage_admis), lorsque le modèle analyse un visage, si la prédiction renvoie le bon label, en d'autres termes si le modèle pense que le visage analysé est le bon visage, alors le programme initie une variable qui va compter le nombre de frame qui s'écoule (grossièrement le temps, d'où le nom de la variable

“tmp”). Si à un moment donné, le visage analysé n’est plus le bon visage, la variable temps est remise à zéro, le système doit détecter le bon visage au moins 45 frames (réglables évidemment) pour être considéré “Admis”. A ce stade, le programme effectue une courte pause pour marquer la détection puis se coupe et renvoie “Admis” dans le terminal. Ceci afin de filtrer en partis les parasites, les faux positifs et les instabilités (en d’autres termes, le programme doit être un peu plus “sûr de lui”). En revanche, si le traitement du visage renvoie le mauvais label, le programme le fait savoir et continue de tourner jusqu’à détection du bon visage ou jusqu’à interruption manuelle.

Ainsi, cette algorithme a encore une fois pus être implanté avec une variété de modèles différents :

- KNN : “3_algo_KNN_Admis_NonAdmis.ipynb”, il est conseillé d’utiliser cette version comme référence
- Arbre de décision : “3_arbre_decision.ipynb”
- GNB : “3_GNB.ipynb”
- Perceptron : “3_Perceptron.ipynb”
- Régression logistique : “3_regression_logistique.ipynb”

Cependant, toutes ces versions partent du postulat que les visages “Non Admis” sont tous des visages stocké dans le dataset, ce qui signifie que les programmes fonctionnent ainsi :

- lorsque confronté avec le visage enregistré dans le dataset et désigné comme **Admis** => Admis
- lorsque confronté avec un visage enregistré dans le dataset et qui n’est **PAS désigné comme Admis** => Non Admis
- lorsque confronté à un visage qui n’est pas enregistré dans le dataset => dûs à la nature du programme, celui-ci assimile le visage inconnus au visage le plus “proche” (classification) => Non Admis ou Admis selon le visage auquel il est assimilé

Ce dernier point peut être considéré comme problématique, c’est pourquoi nous avons également créer une version qui n'assimile pas le visage inconnus à un visage connus suffisamment proche:

- “3_KMeans_autre_methode.ipynb”

Cette version est toutefois plus instable que les précédentes. Son fonctionnement est assez simple. On donne le nom du visage à reconnaître dans la variable “nom_visage_admis”. Puis le programme ne charge uniquement les photos du visage désigné. Ensuite le modèle étant un Kmeans, donc du clustering avec un entraînement non supervisé, cela nous permet de créer un premier cluster uniquement du visage souhaité, puis un second qui sera de tous les autres visages autre que celui souhaité.

Cependant, une meilleure méthode aurait été de modifier le fonctionnement du modèle afin d'inscrire une variable permettant de mesurer la déviation du visage traité par rapport aux visages enregistrés et de considérer que si cette déviation est supérieur à une valeur choisis, alors considérer le visage comme inconnus (en d'autres termes si le visage est trop différent de tout ce qui est enregistré, alors le considérer comme inconnus).

Exercice 4:

L'exercice 4 reprend donc les consignes de l'exercice 2 et 3 mais cette fois-ci en utilisant un modèle de réseau de neurones. Nous avons donc recherché des informations sur ces réseaux dans nos cours avec Madame Emad et vous-même, ainsi que sur le web. Il en est ressortie une très grande ressemblance avec un perceptron qui en est sa forme la plus simple. Suite à ces recherches, nous avons donc essayé d'intégrer un Multi-Layer Perceptron (MLP) de sklearn dans nos codes. Ce "MLP" nous permet assez simplement de transformer nos codes en important ce modèle depuis sklearn. Lors de sa définition, nous avons choisi un modèle de réseau en 5 couches. La première couche possède 100 neurones, la deuxième 80, la troisième 60, la suivante 40 et la dernière 20. On observe que les prédictions sont plus "stables" / meilleures que lors de l'utilisation des autres modèles.
Nous avons utilisé le solver "LBFGS" qui, comme nous avons pu voir sur internet, nous permet un apprentissage plus rapide avec un petit dataset. En effet, ce solver converge plus rapidement que les autres solveurs.

Exercice 5:

Si nous avons bien compris, un réseau de neurones convolutifs utilise plusieurs réseaux de neurones différents traitant chacun une information différente issue du même échantillon (par exemple 3 degré de couleur issue de la même image) puis donnent en entrées leur résultat à un même dernier réseau de neurones qui se charge de la prédiction finale. Nous n'avons cependant pas eu le temps de mettre en place cette méthode.